intel®

APPLICATION NOTE

AP-278

October 1986

Integrating the PC AT Into the Intel Development Environment

SRIVATS SAMPATH
DSO APPLICATIONS

Order Number: 280272-001

INTRODUCTION

In recent years the Personal Computer has become a popular vehicle for delivering computing power to the engineers. IBM's latest offering the Personal Computer AT (Advanced Technology) incorporating INTEL's 80286 16-bit microprocessor, brings about a high level of technical sophistication into a personal computer. The power, speed and memory addressability of the 80286 microprocessor is now available to the user to do tasks which at one time could be run only on a mini or mainframe.

Intel has recognized this growing trend and has introduced translators, debuggers and networking for the PC AT. The same tools that have in the past, been offered only on Intel's proprietary development systems are now available on the PC AT under PC-DOS 3.0 or greater. Language translators are available for the complete spectrum of Intel microcontrollers (MCS®-51 and MCS-96 families) and microprocessors (8086, 80186, 80286). 80386 tools will be introduced early 1987. This is the first time powerful software debuggers like PSCOPE and TSCOPE, hardware debuggers like I²ICE™ have been made available on a personal computer. Intel already supports a broad range of workstations which may be networked to form a productive network. This application note discusses the multiple methods in which the PC AT may be integrated into this development environment.

All software discussed in this application note is available (except where listed) in the Network Toolbox, Part No NDS2TLB 2.0, as discussed in Appendix D.

THE INTEL DEVELOPMENT ENVIRONMENT

The Intel development environment consists of iPDS™ Personal Development System, Series-II's, Series-III's, Series-IV's all of which can be operated standalone or networked through a file sever using the NRM (Network Resource Manager). Intel also supports industry standard hosts such as the DEC VAX and now we include the PC AT as a supported host. Figure 1 shows all combinations of the Intel development environment while Figure 2 illustrates the possible inclusion points of the personal computer.

This appliction note assumes that the reader is familiar with the current Intel Development Environmanet. For more information on the Intel Development Envionment please refer to:

1. AP Note Number AP-244 *DJC A Key To Increased Network Productivity*
2. AP Note Number AP-245 *Creating an Efficient HFS*

These application notes are also available in the 1986 DSO Handbook, Order Number 210940.

This application note deals with integrating the PC AT into an existing Intel development environment. The enclosed details will allow the reader to choose the method that best suits the project. This applications note will discuss three methods of data transfer—serial interconnect, media transfer and networking. Of these, serial transfer is the most universal, media transfer is the most straightforward and Ethernet transfer is the most efficient.

SERIAL INTERCONNECTS

The simplest method of integrating the PC AT into the Intel development environment is through serial interconnects. This method is inexpensive albeit slow. Serial interconnects also allows the user the flexibility to use modems to interconnect with systems which are not in the same location. It allows for both terminal emulation and file transfer at speeds of up to 9600 baud. Serial connections have always been a popular method of linking different computers. Unfortunately this has resulted in a variety of serial communication software being developed that are incompatible over different operating systems. Intel recognized this incompatibility and decided to advocate serial communication software that was compatible over a range of operating systems. The KERMIT file transfer protocol developed by Columbia University, addressed all the needs of serial interconnects, and resolved most inadequacies in previously available serial software. It also solved the multiple operating system incompatibility issue. KERMIT, is available for all hosts shown in Figure 1. Note however that all KERMIT implementations are not equal. The specification details a minimal set of and also specifies numerous additional features which may be added. Currently, the ISIS implementation on the iPDS system, Series II, III and Series IV is a minimal set while the VAX and PC implementations are both extensive. The XENIX and iRMX versions are good and being improved. KERMIT is public domain software and cannot be charged for. Versions for the Intel hosts are available from Insite for a small disk copying fee (see Appendix C).

The following paragraphs discuss the different methods of integrating the PC AT using the KERMIT file transfer protocol. Intel, with help from a number of customers presently using our systems, has developed KERMIT software for the following systems:

- iPDS—Serial port
- Series-II or III—Serial port
- Series-IV—Serial port 2
- ISIS cluster board—Directly into the ISIS cluster board

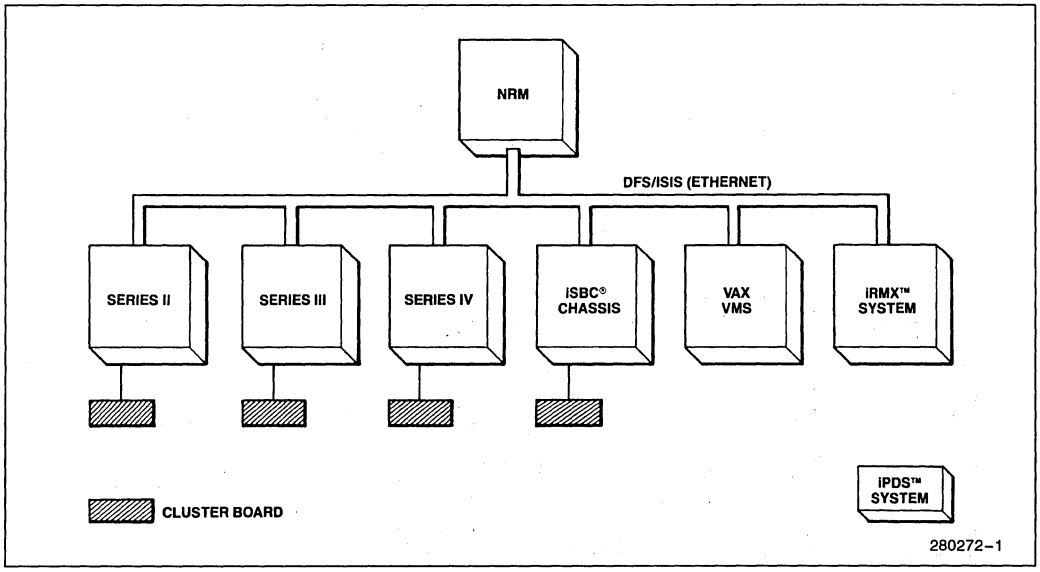

Figure 1. The Intel Development Environment

KERMIT

KERMIT is a file transfer and terminal emulation protocol developed by Columbia University in 1981. Since then KERMIT has been ported to over 30 different systems and is on its way in becoming an industry standard protocol. The KERMIT protocol is designed around character oriented transmission over serial lines. The design allows for peculiarities in transmission medium and requirements of different operating environments. The KERMIT protocol incorporated features and ideas from protocols like DIALNET, DECNET and APPANET. Currently KERMIT has been implemented in over 26 systems. A detailed discussion on the KERMIT protocol is covered in Appendix A.

The following list shows the different systems and operating systems that support the KERMIT protocol.

System	O/S
Series-II, III, IV, iPDS	ISIS
IBM 370 Series	VM/CMS, MVS/TSO, MTS
CDC Cyber	NOS
DEC VAX-11/7XX	VSM, UNIX
PC	MS-DOS, PC-DOS
Apollo	Aegis
PRIME	PRIMOS
HP 3000, 1000	
Apple 11 6502	Apple DOS

KERMIT is a two ended protocol. It needs the remote system to have KERMIT running on it too, to do file transfers. The KERMIT executing on the PC is MS-KERMIT and the one on the Series-II, III, IV, iPDS is the ISIS-KERMIT. The following chapters will detail how this serial interconnect is established.

KERMIT can communicate over either port on the PC AT. Switching between the PC ports can help the PC user communicate with two different systems alternatively.

MS-KERMIT

MS-KERMIT is a program that implements the KERMIT file transfer protocol for the IBM PC AT and several other machines using the same processor family (Intel 8088 or 8086) and operating systems family (PC-DOS or MS-DOS 2.0 or greater).

MS-KERMIT has an extensive command set. A brief summary is shown in Figure 4 wih a more detailed explanation in Appendix A.

ISIS KERMIT

ISIS KERMIT is a minimal KERMIT implementation. This is also available in Insite as described in Appendix C.

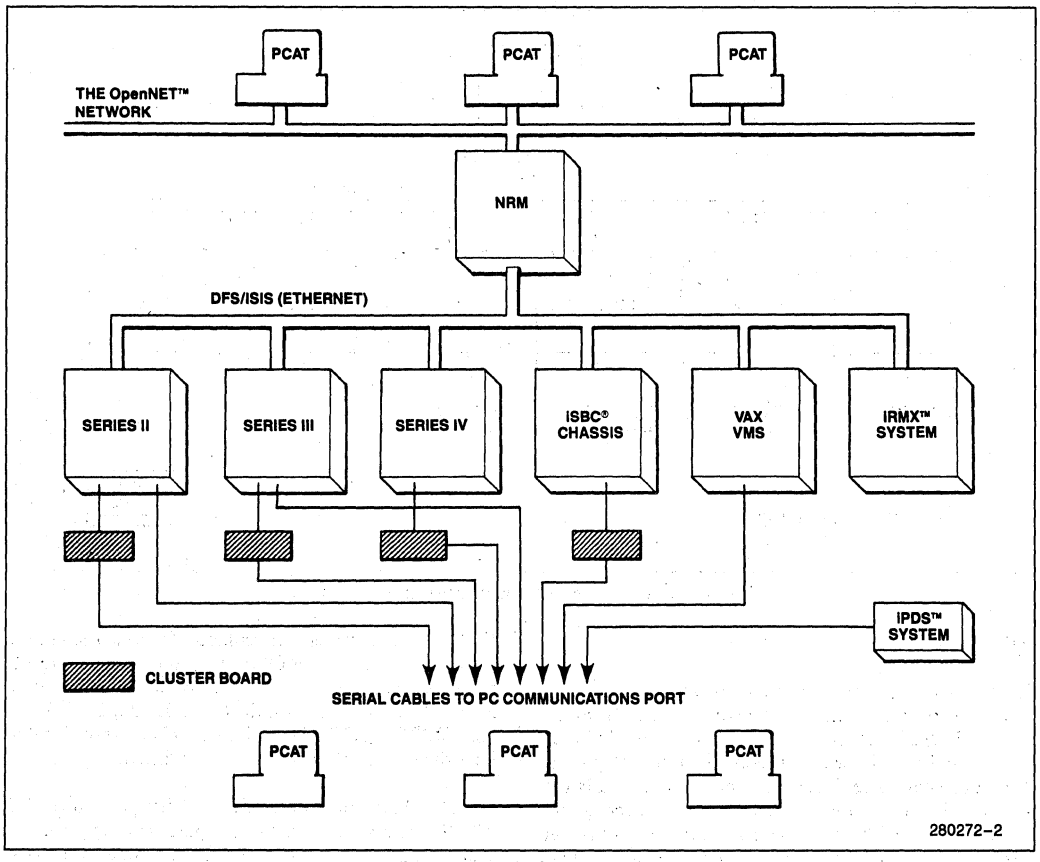

Figure 2. Including the PC

It operates under the ISIS operating system. The basic command set supported by ISIS KERMIT are:

CONNECT—enters terminal mode for communication with host.

DEBUG—toggles debug mode on/off. Prints messages during transfers. Normally used only during trouble-shooting.

EXIT—Return back to ISIS.

SEND filename—specifies the file to be transferred to host. May use the ISIS :fn: drive designation to open file on any logical drive in the system. That drive designator will be stripped from the name before it is sent to the host.

RECEIVE [n]—After commanding the host to send a particular file, or a group of files (wildcards can be used on hosts if they're smart enough), press 'HOME' or 'control]' to drop back to ISIS-KERMIT and enter the 'RECEIVE' command. If the command is followed by a number (0–9) the files(s) will be sent to that logical drive. For example, 'REC 4' will cause the filename(s) to be prefixed by :f4: when opened. The number of drives varies, depending on which system is used.

Since there are only 5 commands, a single letter is all that is required to use them.

'r 3' is euivalent to 'RECEIVE 3'

KERMIT is invoked as follows:

KERMIT [baud-rate] [port number]

The default baud rate is 2400, Others available are 300, 1200, 9600.

The port number selection is effective only on Series II. The iPDS system has only one port, and the Series IV must use to port 2, since it is global in multi-user mode.

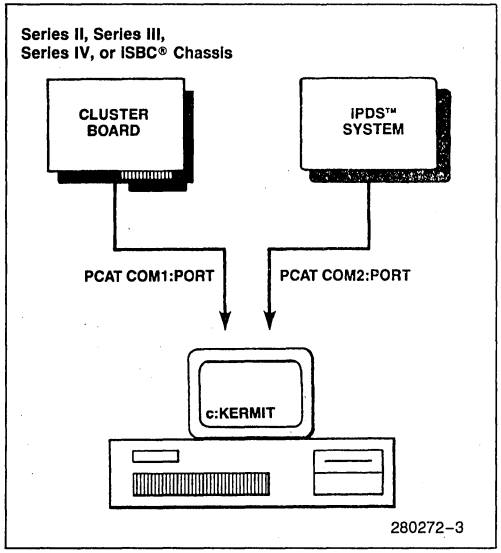

Figure 3. How to Connect the PC to a Remote Host Using KERMIT

A TYPICAL KERMIT SESSION

With the availability of 8051 and 8044 languages on DOS, an existing user may need to move existing software from the iPDS system to the PC AT. The following paragraphs serve both as an example, and as a method to help the iPDS user set up his serial interconnects to do the migration. The various steps, which are explained in detail, include setting up the iPDS system for serial communications, setting up the PC and the actual terminal emulation and file transfer sequence. The steps illustrate the ease with which this migration is brought about.

Command	Explanation
CONNECT	To connect as a remote terminal to a remote system.
DELETE	Delete local files
LOCAL	Prefix for local file management commands
RECEIVE	Receive files from remote system
SEND	Send files to remote system
QUIT	Quit from MS-KERMIT
RUN	Execute a MS-DOS program
SET	Set parameters like baud rate, serial channel
SHOW	Display all parameters
EXI	Exit from MS-KERMIT
DIRECTORY	Directory of local PC

Figure 4. KERMIT Command Set

STEP 1.

Install ISIS KERMIT on a iPDS diskette, and create a CSD file ABOOT.CSD that looks like this:

```
SERIAL A    B = 9600
  ;Set the serial port in ASYNC mode at 9600
ASSIGN :CO:TO :SO:
  ;Redirect console out to the serial port
ASSIGN :CI:TO :SI:
  ;Redirect console in to the serial port
```

This step sets up the iPDS for serial communication by initializing the serial port to communicate asynchronously at 9600 baud. The console I/O redirection is done to enable KERMIT-MS to control the iPDS. Placing these commands in the ABOOT.CSD file help bring up the iPDS system in the right mode whenever it is reset.

STEP 2.

Install DOS KERMIT on the PC AT and invoke it by typing KERMIT from the command line.

```
C: \> KERMIT

IBM-PC KERMIT-MS VER 2.26
TYPE? FOR HELP
KEMIT-MS > SET BAUD 9600
KERMIT-MS >connect
```

Once a successful connection has been made to the iPDS the PC AT terminal will display the iPDS prompt.

```
A0>
```

Now the user can do any operation like DIR, ASSIGN, etc., on the iPDS system from the PC keyboard.

STEP 3.

```
FOR FILE TRANSFER.
```

Invoke the iPDS KERMIT by typing in KERMIT

```
A0>KERMIT 9600 1 ;9600= baud rate and
1=port
```

The ISIS KERMIT prompt will appear ISIS-KERMIT>

For receiving files type in

```
ISIS-KERMIT>RECEIVE :F0:
```

Exit back to the PC by typing in CNTRL] C at the same time. The user is now back to the KERMIT-MS> prompt. Now type in

```
KERMIT-MS>SEND EXAMPLE.BAT
```

A Screen comes up showing data transfer status and on successful completion on file transfer will give back the prompt. More information on setting the number of retries on error packets and timeouts are explained in Appendix A.

```
KERMIT-MS>
```

MEDIA TRANSFER UTILITIES

Diskettes constitute the main source for data and information storage. Most software is kept on diskettes for ease of storage, transportability, and safekeeping. Migrating from one host system to another involves transferring this data on to the new host system media. Media transfer is useful only if both hosts are at the same site and support a compatible peripheral device. If both these conditions are met, media transfer provides a fast convenient way for casual data transfer.

Handling diskettes and interacting with two host computer systems is error-prone and inconvenient. I recommended the other two methods of file transfer in a production environment where the two computers may converse without operator intervention. One major problem with media transfer is the lack of industry standards. There is an 8 inch single density standard (IBM 3740) but not for other densities nor for 5¼ inch media. To solve this problem, special host dependent utility programs must be written to permit the reading of another systems diskettes. This was addressed for the PC by developing a set of utilities that allow file transfer from 5¼ inch and 8 inch. This gives the user the ability to move between the Series-IV environment and the PC-DOS environment with the least overhead and loss of productivity. A key factor in projects these days.

MSCOPY is program that manipulates a MS-DOC disk on a Series IV or NRM. It also helps the PC AT user access the NRM print spooler. The user can copy software both to and from a Series-IV. MSCOPY expects the MS-DOS diskette in FL0. While running MSCOPY do NOT change the disk as MSCOPY keeps the Disk allocation table in memory and will not reread them from a new disk but will write out the old table and directory.

MSCOPY supports 48 or 96 TPI disks, 8 or 9 sectors per track, 1 or 2 heads, MS-DOS vers. 1, 2 or 3. It does not support 1.2 Mb high density diskettes.

It has two modes of operation, interactive and non-interactive. In the non-interactive mode you may enter only one command and it must deal only with the MS-DOS root directory. To use the non-interactive type the command on the invocation line.

In the interactive mode, (entered by invoking MSCOPY with no parameters) MSCOPY will prompt you for a command. Currently there are seven legal commands.

The seven commands are:

READ msfile indxfile—Copies msfile to indxfile. msfile must be in the current directory. indxfile can be any valid iNDX pathname up to 40 characters long.

WRITE indxfile msfile—Copies indxfile to msfile. msfile will be added to the current directory. indxfile can be any valid iNDX pathname up to 40 characters long.

CD msdir—Changes the current directory to the directory msdir. This command will only go up or down the tree one node at a time. To go back one level say "CD". To go deeper say "CD name" where name is a dir entry in the current directory. Typing in "CD \" will jump to the root directory.

DELETE msfile—Removes msfile from the current directory and reclaims the space it occupied.

RELAB label name—Will change or add the volume name of the MS-DOS disk. Label name may be up to eleven characters long.

DIR—Will display the current directory.

EXIT—To return to iNDX.

8" ISIS Media

Flagstaff Engineering in Phoenix, Arizona have a set of tools that allow direct transfer from 8" ISIS (SS/SD) media to PC's. The tool set consists of a add-on board for the PC, an 8" drive and the driver software to do the required transfer. File transfer is bidirectional. The program ISS8TO5 copies files from 8 inch media to PC, and ISS5TO8 copies the other way. An example is shown in Figure 5.

Please note that Intel does not sell, support or warrant reliability of this product. Intel's evaluation sample has proved reliable and Flagstaff technical support has been good.

For more information please contact:

Flagstaff Engineering
Box 1970
Flagstaff, AZ 86001

```
>ISS8T05
COPY INTEL ISIS DISKETTE FILE TO IBM PC-DOS FILE PROGRAM
COPYRIGHT FLAGSTAFF ENGINEERING 10/17/83

THIS PROGRAM WILL COPY A FILE FROM A 8" ISIS SINGLE DENSITY DISKETTE TO AN IBM
PC-DOS DATA FILE. THE FILES MUST BE CREATED USING ISIS-II OR RMX/80 SYSTEMS.

INSERT 8" ISIS DISKETTE--ENTER DRIVE(1/2) WHEN READY.?
FILE DIRECTORY FOR DISKETTE 164539001
01-ISIS    .DIR(025)  02-ISIS    .MAP(002)  03-ISIS    .TO(023)
04-ICE51   .   (253)  05-ICE51   .0V0(020)  06-ICE51   .0V1(011)
07-ICE51   .0V3(027)  08-ICE51   .0V4(008)  09-ICE51   .0V5(036)
10-ICE51   .0VE(082)  11-ICE51   .0VH(498)  12-ICE51   .0VS(049)

ENTER ISIS FILE NUMBER (1-96/99=ALL)--PRESS ENTER IF NONE?
DO YOU WANT TO COPY FROM ANOTHER ISIS DISKETTE (N/Y)?
```

Figure 5

NETWORKING THE PC AT WITH THE OpenNET™ SYSTEM

OpenNET™ Architecture

OpenNET is Intel's Local Area Network architecture. OpenNET conforms to the Open Systems Interconnect (OSI) model defined by the International Standards Organization (ISO). The major objective of ISO is to create an open systems networking environment where any vendor's computer system can be connected to any network and freely share data with the network.

The OSI ISO architecture is based on a seven layer model (see Figure 6). The seven layers isolate independent functions so that the network can better make use of new software and hardware without adversely affecting the other layers. The upper three layers (5 through 7) provide interoperation functions, while the lower four layers (1 through 4) provide interconnect functions and the bottom two layers (1 through 2) are concerned with the transmission through physical medium.

OpenNET™ Family

As part of Intel's Open Development Environment (ODE), OpenNET supports a number of industry standard hosts and operating systems. To date, OpenNET runs on the IBM PC family with PC-DOS 3.1 or greater, iRMX, XENIX and iNDX as shown in Figure 7.

Since all of the above mentioned systems conform to the ISO seven layer model, they can all interoperate and interconnect over the same network.

OpenNET™ PC Link: Overview

Intel's PC connection on the OpenNET system, named OpenNET PC Link, consists of an add-in controller board for the PC, XT or AT (Layers 1–2), the iNA960 ISO transport software (layers 3–4) and the MS-NET software (layers 5–7).

PC AND THE NRM FILE SERVER ON THE OpenNET™ SYSTEM

The remainder of this application note discusses the use of a PC on the OpenNET system with Intel's Nework Resource Manager (NRM) as the OpenNET file server.

The current NDS-II NRM can be converted into an OpenNET file server by installing the iSXM™ 552 board and iNDX R3.0 or greater software (the board and the software have been kitted into the "NDS-II OpenNET Upgrade Kit", Part # "iMDX555"). New users have the choice of a Mini OpenNET NRM with a 40 Mb disk or a Maxi OpenNET NRM with a 140 Mb disk (upgradeable to 4 140 Mb disks) and a 60 Mb tape.

DETAILED EXPLANATION OF CONNECTING PC TO THE OpenNET™ SYSTEM

The OpenNET system uses concepts such as SERVERS and CONSUMERS which allow a building block approach to creating a network that can be tailored to your particular specification. The following chapters will discuss indepth the various concepts of the OpenNET system and on how to implement PC's and iNDX systems on an OpenNET network.

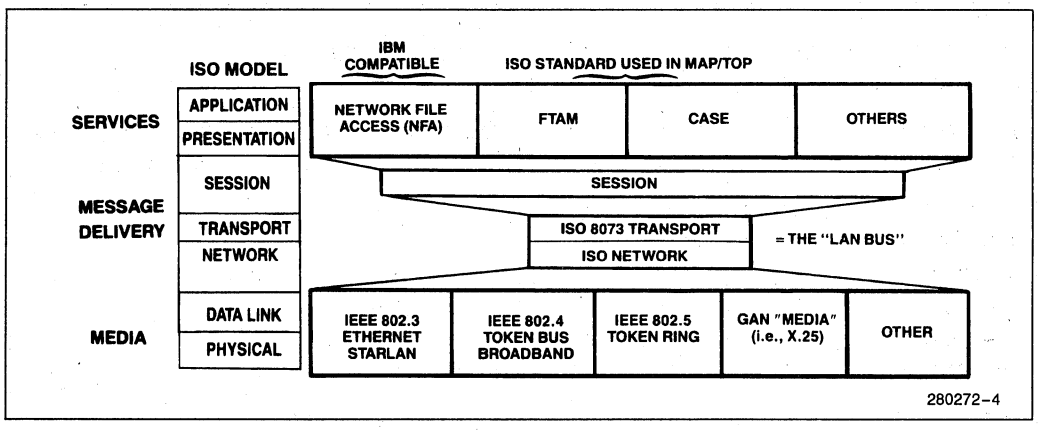

Figure 6

A server is defined as a system on which network resources like files, directories and a printer are kept. A Server usually has a number of hard disks. It is called a "SERVER" because it serves the other systems on the network when they request for files and printer service. There may be a number of servers on the network. The NRM with the iSXM 552 board and iNDX 3.0 installed in it acts as a SERVER for the other systems on the OpenNET network. XENIX and iRMX system can also be servers.

Computers that are linked to a server and use it as a resource for files and printer service are called CONSUMERS. CONSUMERS can also operate independent of a SERVER. An example of a CONSUMER on the OpenNET network is the PC. It can operate independently as a workstation and also uses the NRM for file services. XENIX and iRMX are capable of operating as consumers too.

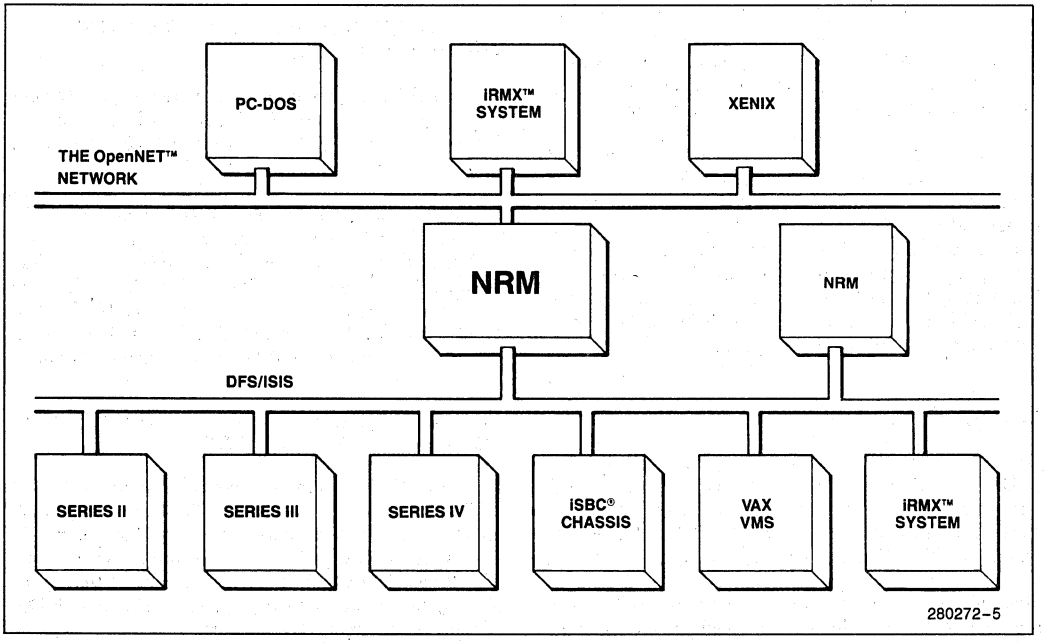

Figure 7

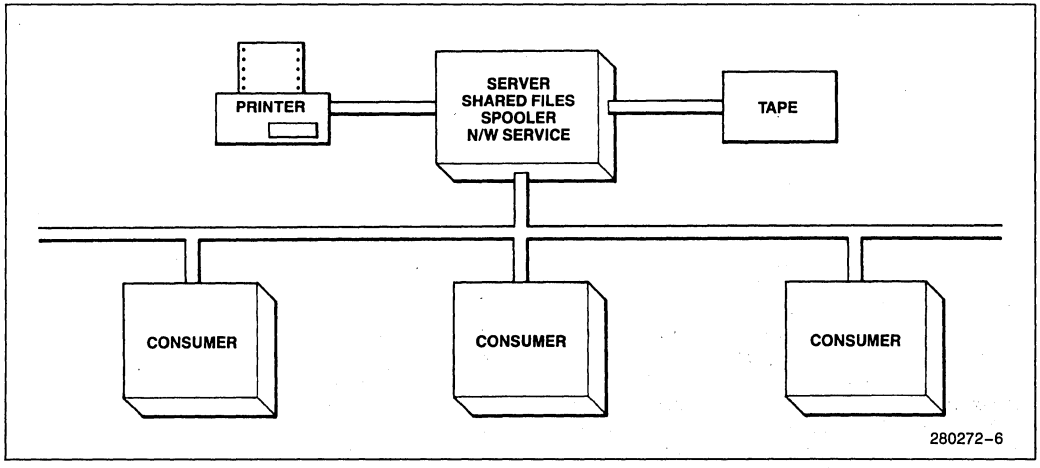

Figure 8. Server and Consumer

A list of all servers and consumers on an OpenNET system is stored in a database file called the NETADDR file. This is discussed in Appendix C.

Figure 8 illustrates how the server and consumers interact on a network.

A SAMPLE OpenNET™ SESSION FROM THE PC TO THE NRM

The following paragraphs will describe how the PC user can access files at the NRM. But before going into the details a few hints on making the process automatic and easy.

Since the OpenNET system uses the concept of virtual drives, it will be beneficial to have as many virtual drives as possible. Refer to the Virtual Drives section under the Chapter "Connecting PC's to OpenNET". DOS 3.1 has a default number of virtual drives 5 (A: through E:), however for OpenNET more drives may be needed. This can be achieved through modifying the CONFIG.SYS file in the root directory of the PC. Edit this file to include the command:

```
lastdrive = z  ;increase the number of
               virtual drives to 26.
```

A typical CONFIG.SYS file is shown in Figure 9.

On boot up, DOS 3.1 will read this file and automatically configure the PC as specified.

Now start the PC as a consumer by entering:

```
C:NET START RDR <this PC name>
```

This is specified in the NETADDR file discussed in Appendix C.

This command loads the PC-LINK communication software onto the PC-Link board, and sets up the environment for communicating with any server. This sets up the session layer on the controller board.

If all the steps went through successfully, the network software will be loaded into the PC-Link board and will sign on with:

```
*****************************************
*                                       *
*            OpenNET™ PC Link           *
*                                       *
*     Copyright 1985, Intel Corporation *
*                                       *
*****************************************

C:>
```

Now connect to the NRM using the NET Use command. The syntax for this command is:

```
C:> NET USE <virtual drive>\\<server name>
 \username password
```

Example:

```
C:> NET USE J:\\APPS--NRM1/GUEST WELCOME
```

The above command creates a virtual drive J: which points to the home directory of the user GUEST with a password WELCOME at APPS—NRM1. This drive J: is like any PC drive except that it points to a directory

```
lastdrive = z           ;Set the number of virtual drives to 26
device = \sys\ansi.sys  ;Set the terminal characteristics to ANSI
files = 20              ;Number of files open at one single time
buffers = 20            ;Number of buffers for file I/O
break on                ;set break key on
```

Figure 9

at a remote NRM. The user can do any DOS function like copy, dir, etc. even execute DOS applications programs that are stored at the NRM in this directory. Just by having this capacity the PC becomes a very powerful and flexible workstation. By sitting at one PC the user can have access to several file servers.

The command will come back with a message "command successfully completed" if it was successful, otherwise it will wait about 4 minutes before timing out and giving back the DOS prompt.

The user can now use this virtual drive just as if it was any other PC drive. For example a DIR command on drive J: will look like this.

```
C:>dir J:

Volume in drive J has no label
Directory of J:/

INIT        BAK          83   9-06-85    9:23a
INIT        CSD         290   9-06-85    9:23A
CDISK       CPM      401408   9-06-85    9:24a
NNMAC       MAC          74   9-06-85    9:24A
HILIB            <DIR>       9-06-85    9:26a
DJC              <DIR>       9-06-85    9:26a
DATABASE    DIR  <DIR>       9-06-85    9:55a
KERMIT      DIR  <DIR>       9-06-85    9:57a
IMPORT      DIR  <DIR>       9-06-85   10:15a
OPENNET     DIR  <DIR>       9-06-85   10:28a
     10 File(s)   30642176 bytes free
```

This is the home directory of the user GUEST at the NRM. Users familiar with the Intel development environment, will notice that the OpenNET network translated the output of the DIR command at the NRM to DOS format. The user can change directories at this drive, invoke DOS applications from this drive, if they are stored at the NRM, store data files on this drive. The underlying OpenNET protocol is transparent so all existing DOS applications programs can access this drive just as if it was stored locally.

Any time a user wants to fiind out which of his virtual drives are connected to which servers he enters the NET USE command.

```
C:> NET USE
Local Network
Status Device Name
---------------------------
E:    \\APPS_NRM1\GUEST
F:    \\APPS_NRM2\GUEST
G:    \\APPS_NRM1\GUEST
Command completed successfully.
```

More information on the NET USE commands are given in the OpenNET PC Link manuals.

DISTRIBUTED JOB CONTROL SYSTEM FOR THE PC

The Distributed Job Control system on the NDS-II allows currently idle networked development systems to be supplied to the network as public resources. This is a remote job execution unit to which jobs can be sent by other users on the network. Remote job execution offers higher throughput and increased efficiency, as more than one computer can be controlled by a single user. For more information on DJC, refer to Application Note 244, "DJC A key to increased network productivity". It is recommended that this Application Note be read since a number of concepts explained in the following paragraphs assumes that the user has knowledge of the DJC system at the NRM.

The Network Resource Manager (NRM) is the nerve center of the DJC system. All jobs are scheduled and queued by the NRM. Traditionally the PC user had to wait for his compiles to be finished locally before doing anything else on the PC. With the introduction of the OpenNET network, a mechanism has been designed to let the resources of DJC at the NRM be made available to the PC user. This feature enables the PC user to edit files at he PC and send the compiles over to the NRM. An efficient tracking system has also been designed to help keep the user informed at all times on the status of his job. With the introduction of the 286/310 iNDX based Compile Engine shown in Figure 10, the

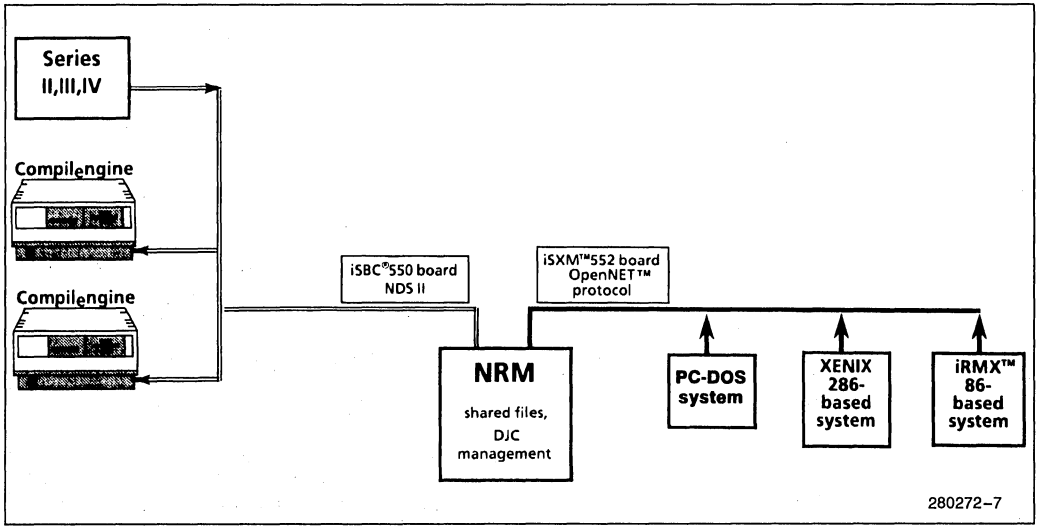

Figure 10. PC's, Compilengine, DJC and the OpenNET™ Systems

```
Reexporter          ;invoke the REEXPORTER utility
Export Reexport to iNDXUTILITY.Q nolog ;now export this job again
```

Figure 11. Contents of REEXPORT.CSD

throughput on such exported jobs increases dramatically. This directly translates into increased productivity and efficiency for the PC user.

The PC-Export package consists of a utility that runs at an iNDX station on the DFS side of the NRM (i.e., NRM itself, Series-IV or 286/310 compile Engine). This package called REEXPORTER.86 checks through a specified directory of all users on the network and if any jobs from the PC exist it will export it to the appropriate queue at the NRM and then will delete the file. Each OpenNET PC user who needs access to the DJC manager at the NRM must create a directory NRMDJC.DIR under his/her HOME directory. It is advisable to limit the above operation to only those users who need to access the DJC system at the NRM. This will help the REEXPORTER utility in having a faster turnaround rate, by not checking redundant directories.

The Superuser then has to create an export file REEXPORT.CSD shown in Figure 11.

It is assumed that the NRM has three queues, 8bit.q (for 8 bit jobs), 16bit.q (for 16 bit jobs) and iNDXUTILITY.Q (for utilities other than compiles, limited to doing system administrative jobs). For more information please read Application Note #244 "DJC a Key to increased network productivity". The Superuser must bring up a Series-IV workstation, or a 286/310 Compile Engine in Import mode, importing from all the three queues. This is shown in Figure 12.

```
>Import from 8bit.q, 16bit.q, indxutility.q
```

Figure 12

The command now puts the workstation as a network resource that all user can acess.

Now export REEXPORT.CSD to iNDXUTILITY.Q

```
>Export REEXPORT.CSD to iNDXUTILITY.Q Nolog
```

Exporting from the PC

To export jobs from the PC, the user must first connect to the NRM using the NET USE command explained in previous chapters. One of the design considerations was to make this utility as easy to use as possible. The following paragraphs illustrate how this has been brought about.

Consider this example file that does a compile and link, COMPILE.CSD. A requisite is that the file must have a .CSD extension, as it is this extension that informs the NRM that it is a command file.

```
;queue = "16bit.q"
lname define l for/wini0/libs.dir
lname define p for /wini0/strng.dir
plm86 example.p86 debug optimize (2)
if % status = 0
 link86 example.obj,l/compac.lib,&
   p/hstrng.lib,l/osxcom.lib &
   to example.86 bind
 end
```

The first line in the command file is a comment, which also indicates the queue where this job is to be executed. As far as the DJC manager is concerned, this is just a comment. But the REEXPORTER utility uses this field to find out the queue name. This comment field must exist within the first 128 bytes of the command file. An absence of this field will result in the job not being sent to any queue. The queue name must be enclosed within double quotes.

Since there is no way by which the NRM can access files stored on the PC, all source, libraries and objects must reside on the network. Note the two commands after the comment which set up the logical names for directories used in the job. Doing this will ensure that the right libraries are used.

Now all the PC user has to do is to copy this file to the directory NRMDJC.DIR under their home directory at the NRM. The REEXPORTER utility does the rest. For example if virtual drive G: has been connected to the NRM.

```
C: \Copy COMPILE.CSD G: \NRMDJC.DIR
```

Once the job has been reexported it will be deleted from the directory. This helps the user in determining if the job got exported or not. The REEXPORTER utility also creates a log file in the NRMDIC.DIR directory for each job exported. This allows the user to find out if his job was successful or not. The COMPILE.CSD file will be replaced by a COMPILE.LOG file once the job has been completed. The COMPILE.LOG file is a LOG file of all the operations by the job.

The REEXPORTER utility was designed and implemented to allow the OpenNET PC user access to the powerful Distributed Job Control mechanism at the NRM. The utility has the capability to determine all the users on the network, work out their home directories and look for jobs sent from PC's. On finding a job, the utility determines the appropriate queue and reexports the job to that queue. The status of each job is displayed on the screen at all times. Status information includes username, jobname, queue and the status of the exported job. The use of this utility is restricted to the Superuser. A normal user invoking REEXPORTER.86 will generate an insufficient access rights exception.

REEXPORT.CSD is a batch file configured as a job that runs forever. It first uses REEXPORTER to check for jobs in the directory NRMDJC.DIR of all users and exports them to appropriate queues. It then reexports itself to the same queue. Due to the way the DJC mechanism is structured, the import station will start executing all the jobs found by the REEXPORTER utility, and on completing all of them will execute the REEXPORT.CSD job once again to look for more work to do. The cycle keeps repeating forever.

Referring back to the IMPORT command in Figure 12, highest priority is given to 8bit.q and lowest to iNDXUTILITY.Q. This way the system manager makes sure that all jobs waiting in the first two queues are executed before the REEXPORT.CSD job is started again. This helps the NRM in controlling the queues, and not overloading them at one single time. A point to note at this time is that the REEXPORTER utility is capable of exporting up to 23 jobs a minute.

The REEXPORT utility combined with OpenNET networking opens out a completely new environment for the PC AT user. An environment where compiles, links and locates can be done remotely. The PC user has at his/her disposal the power of the iNDX Distributed Job Control subsystem. This help in bringing about an increase in productivity that normally could not have been achieved without Intel's OpenNET network. The PC AT user can spend more time on interactive work such as program generation or debugging, while the compiles are being done elsewhere. This feature set should be used by developers doing system designs in todays world where "Time to Market" is key.

Summary

This application note has discussed in detail all the different methods by which the PC AT can be integrated into the Intel Development Environment, from serial interfaces to networking. These different methods can be intermixed to suit your needs. Serial interfaces and disk transfers support the low end needs while OpenNET brings about a powerful new environment to the PC AT. This coupled with the REEXPORTER utility, can help increase the productivity of the PC AT user dramatically.

APPENDIX A
THE KERMIT PROTOCOL

THE KERMIT PROTOCOL

The KERMIT protocol is designed around character oriented transmission over serial lines, and incorporates features from decnet, arpanet, dialnet etc.

File transfer takes place over transactions. A transaction is an exchange of packets. A successful transaction is done when one system sends a packet and the remote system acknowledges it.

Transmission begins with a send_init packet(s) and ends with a break_transmission (b) or error (e) packet. All communication is done through packets, even if no data is being sent.

The Kermit packet is built around the following format.

mark	length	seq	type	data	check

All the fields are ASCII characters.

Mark

This is the synchronization character that marks the beginning of a packet. This is normally a cntrl-a and can be redefined.

Length

The number of ASCII characters within the packet following this field.

Seq

The packet sequence number, ranging from 0 to 63. Sequence numbers wrap around to 0 after each group of 64.

Type

The packet type is represented in a single ASCII character. The different packet types are:

```
D data packet
Y acknowledge
N negative acknowledge
S send initiate
B break transmission
F file header
Z end of file
E error
```

Data

The contents of this packet.

Check

A checksum on the characters between, but not including the mark and check. The check for each packet is computed by each host and must be equal for the packet to be accepted.

KERMIT File Transfer Sequence

File transfer is initiated by the sender sending a send_initiate packet, where parameters like packet length, time out limits are specified.

The receiver than sends an ack (y) with its own parameters in the data field.

The sender then transmits a file—header packet which contains the filename in the data field. The receiver then sends an ack.

The sender then sends the contents of the file in data packets (d), any data that is not in the printable range is prefixed and replaced by a printable equivalent. Each d packet has to be acknowledged before the next one is sent.

After all the file data has been sent the sender then sends an eof packet. The receiver acks it.

End_of_transmission packet (b). The receiver acks it and the transaction is over.

KERMIT-MS Commands

CONNECT

The CONNECT command connects the PC as a terminal to the remote system. KERMIT-MS uses either communications port 1 or 2 and uses full duplex and no parity. These can be changed using the SET command. To get back to KERMIT from terminal emulation type in the escape character followed by the letter C. The escape character by default is CENTRL-]. This can be modified by the SET ESCAPE commnd. SET BAUD changes the baud rate, SET PORT changes the serial port. For example:

```
C:\KERMIT <cr>
KERMIT-MS>SET PROT 1    ;select port 1
KERMIT-MS>SET BAUD 9600
KERMIT-MS>C             ;connect
```

SEND

The SEND command causes a file or a group of files to be sent from the local PC to the KERMIT on the remote system. The remote KERMIT must be running in either server or interactive mode; in the latter case the user must have already given a RECEIVE command and escaped back to the PC.

SEND filespec1 [filespec2]

If filespec1 contains a wildcard then all matching files will be sent in the same order that the DOS would show them on a directory listing. If filespec1 contains a single file, the user may direct KERMIT-MS to send that file with a different name.

SEND KERMIT.ASM TEST.ASM would send the file KERMIT.ASM as TEST.ASM

or

SEND *.ASM will send all files with the extension .ASM to the remote system.

Once the SEND command has been invoked the name of each file will be displayed, packets transferred, retries and other counts will be displayed along with other informational messages. If file transfer is successful a "COMPLETE" message will be displayed else an error message will be displayed. When the specified operation has been done the program will sound a beep.

Several single character commands can be given while a file transfer is in progress.

CNTRL-X	Stop sending the current file and go on the next one.
CNTRL-Z	Abort file transfer.
CNTRL-C	Return to KERMIT-MS.
CNTRL-E	Send an ERROR packet to the remote server in an attempt to bring it back to server or interactive mode.

RECEIVE

The RECEIVE command tells KERMIT-MS to receive a file or a group of files from the remote KERMIT. KERMIT-MS simply waits for the file or files to arrive. The user should have already issued a SEND command at the remote KERMIT and escaped back to the PC before issuing the RECEIVE command.

Syntax:

RECEIVE filespec

If the optional filespec is provided the incoming file is stored under that name. The filespec may include a device designator or may consist only of a device designator. For example:

RECEIVE TEST.ONE	;will name the incoming file as TEST.ONE
RECEIVE A:TEST.ONE	;will name the incoming file as TEST.ONE and store it in drive A:
RECEIVE A:	;will store all incoming files in Drive A

If an incoming file does not arrive in its entirety, KERMIT-MS will normally discard it. This may be changed by the SET INCOMPLETE KEEP command, which will keep as much of the file that arrived successfully.

If the incoming file has the same name as a file that already exists and "WARNING" is set ON, KERMIT-MS will change the incoming file name and inform the user of the new name. If WARNING has been SET OFF using the SET WARNING OFF command, files with the same name as incoming files will not survive.

SET

The SET command allows the user to modify various parameters for file transfer and terminal emulation. These parameters can be displayed with the SHOW command.

APPENDIX B
SETTING UP OpenNET™ CONNECTIONS BETWEEN PC & NRM

APPS__NRM1	:address = 0x80000a0100000000aa00003510000000
APPS__NRM2	:address = 0x80000a0100000000aa000034de000000
APPS__XENIX__1	:address = 0x80000a0100000000aa00002de2000000
DIAG__NRM	:address = 0x80000a0100000000aa00000c0f000000
MFNG__RMX__1	:address = 0x80000a0100000000aa000006e4000000
MKTG__NRM	:address = 0x80000a0100000000aa00000304000000
APPS__PCAT__SS	:address = 0x00010a0100000000dd00002584000000

Figure 13

OpenNET™ CONCEPTS

To enable the PC to talk DFS-OpenNET protocol, the user must install a PC-Link card in the IBM PC, and the networking software PC-LINK supplied by Intel. The prerequisite is that the PC should have PC-DOS Version 3.1 or greater. Follow the installation instructions given in the PC-Link manual. It is advisable to create a directory on the PC called COM (short for Communication S/W) and install all the PC-LINK software in this directory.

Once the hardware and software have been installed on the system the user can now use the PC as a consumer off of the NRM. Before going into a discussion on the actual use, a number of concepts have to be explained, to give the reader a better understanding of how PC's work when networked to the NRM using the OpenNET protocols.

The NETADDR File

Each computer on the network is assigned a name, to identify it. These computers can be named anything, but it is preferable to have one standard naming convention. This makes it easier for the users to connect up to the server of choice. The NETADDR file is a text file that contains the names of these servers and their addresses on the network. This is extensively used by the PC-LINK software to connect to the desired server. This gives the user the flexibility to connect to any server just by giving its name, and the PC-LINK software automatically directs the connection to that server. A sample NETADDR file is shown in Figure 13.

The NETADDR file above has the lists of 6 servers on the OpenNET network. Note the naming convention. The first four letters in the name indicate the department where the server is stationed. For example APPs is short for Applications Engineering, DIAG for Diagnostics Engineering, MFNG Manufacturing etc. The rest of the name indicates the type of server (NRM, XENIX or RMX). This naming convention helps in connecting up to the different servers without any confusion. This file is created by each individual PC consumer using any standard text editor (one which does not put control characters in the file). One of the entries in the NETADDR file is the name of the users PC where this file resides. This is important as the PC-LINK software cannot be invoked without this information. The name and ethernet address of the consumer station is necessary for the NRM OpenNET file server to send message packets back to the correct originating consumer (i.e., the NRM should know which consumer station has requested for a particular process for sending back the response to it).

The rest of the numbers following each computer name are the Port address and the iSXM 552 Ethernet address of the server. An example is shown in Figure 14.

The Ethernet address of the iSXM 552 is obtained from the NRM on boot up. Refer to the Chapter "Installing the iSXM 552 in the NRM" for further details.

The MSNET.INI File

The MSNET.INI file is the PC-LINK initialization file for setting up the help files and loading the PC-Link

APPS	is the department where the server is located
NRM2	is the NRM name (2 is used to differentiate it from the other NRM)
80	is the port address (always 80)
00aa000034de	is the ethernet address of the iSXM552 board at NRM2.

Figure 14

board with the communication software. Any time a PC station is brought up as a consumer this file is parsed and executed. The file may need a little modification if the communication software is located in some other directory. The sample MSNET.INI file is used as an example, see Figure 15.

Virtual Drives

PC-LINK uses the concept of virtual drives to connect to the NRM file server through the OpenNET network. When a connection to a NRM server is made the PC-LINK software creates a virtual drive, which is identical to the PC drives. The only difference is that it does not physically exist on the PC. The user treats this as any other drive on the PC. The number of virtual drives that can be used is limited to the English alphabet (26). The user can connect different virtual drives to different directories at the NRM file server and all these drives can be used just as if they existed on the PC.

Configuring PC Link

PC Link as shipped uses default settings for its memory window, number of connections, interrupt levels and number of servers that it can access. The following paragraphs will discuss various configuration parameters that will allow the user to configure PC link to suit his environment.

CONFIGURING THE BASE FOR THE PC LINK BOARD

All communication with the PC link board takes place via a dual-port memory, which is a 32K window that may start on any 64K window within the first megabyte of addressable memory. The default base is 0A000h. In some cases the window might clash with an existing board or application. This default can be changed by jumpers on the PC link board and by making modifications to the MSNET.INI file. Figure 16 indicates all possible bases and associated jumpers.

After the jumpers have been selected, the MSNET.INI file has to be changed to inform the PC link software about the new window being used. Referring to Figure 15:

```
start redirector $1
start rdr $1
 \command.com/c type \com\pclink.msg
 chknet
 xport/sys:at/base:d
   /file:c:\com\ubcode.mem
 session \com\netaddr
 redir
 setname $1
 \command.com/c psclose
```

The string of commands following the start redirector $1 or start rdr $1 indicate the sequence of events in

```
use $*
   use $* /*

print $*
   printq $*

name
   setname

start redirector $1
start rdr $1
   \command.com /c type \com\pclink.msg
   chknet
   xport/sys:at /base:d
    /file:c:\com\ubcode.mem\nvcs:5
   session \com\netaddr
   redir
   setname $1
   \command.com /c psclose
```

Figure 15. (Sample MSNET.INI file)

loading the PC link board and starting the network. XPORT is the network program that loads the PC link board and the driver. One of the options that can be specified in the xport commands is the base option. In the previous example the base was set to D. This base should reflect the base at which the PC link board is strapped.

Configuring Connection Limits

PC link allows the user to have simultaneous connections to a number of file servers which is configurable.

The PC link defaults allow the user to connect to two file servers simultaneously and have up to 5 active virtual circuits. The MSNET.INI file supplied with PC link has to be modified for users who need access to more than two file servers and more simultaneous connections.

For example:

A user needs connections to three different servers. The first two NET USE commands will come back successfully, however the third command will come back immediately with a message "Connection Refused". This is due to the fact that the PC link software uses a default value of 2 for the number of servers it can simultaneously access.

Starting Address	E5	E8	E11	E14
00000	E4	E7	E10	E13
10000	E4	E7	E10	E15
20000	E4	E7	E10	E13
30000	E4	E7	E10	E15
40000	E4	E9	E10	E13
50000	E4	E9	E10	E15
60000	E4	E9	E12	E13
70000	E4	E9	E12	E15
80000	E6	E7	E10	E13
90000	E6	E7	E10	E15
A0000 (DEFAULT)	E6	E7	E12	E13
B0000	E6	E7	E12	E15
C0000	E6	E9	E10	E13
D0000	E6	E9	E10	E15
E0000	E6	E9	E12	E13
F0000	E6	E9	E12	E15

Figure 16

```
C:\ NET USE H: \\APPS_NRM\JOHN PASSME
Command completed successfully
C:\ NET USE I: \\DEMO_NRM\JOHN PASSME
Command completed successfully
C:\ NET USE LPT1: \\DIAG_NRM\JOHN PASSME
Command Refused
```

In the above case the user tried to access more than one server and since the default was set at 2, the PC link network management facility came back with an error. To change this default value, edit the MSNET.INI file and modify the REDIR specification to read:

```
REDIR /S:5  ;connection available for up to
5 servers
```

Reboot the PC and PC link will now allow the user to connect to up to 5 servers simultaneously.

Another variable that can be configured is the number of active virtual circuits at the PC. As a default PC link will allow up to 5 net uses (Logons) to a server. Again when the default limit is exceeded the system will come back with a "CONNECTION REFUSED" message. For example consider the following NET USES:

```
C:\ NET USE F: \\APPS_NRM\SRIVATS PASSME
Command completed successfully
C:\ NET USE G: \\APPS_NRM\AP1 PASSAP1
Command completed successfully
C:\ NET USE H: \\APPS_NRM\SY1 PASSSY1
Command completed successfully
C:\ NET USE I: \\APPS_NRM\SY0 PASSSY0
Command completed successfully
C:\ NET USE J: \\APPS_NRM\AP0 PASSAP0
Command completed successfully
C:\ NET USE K: \\APPS_NRM\JOHN PASSME
Connection Refused.
```

In the above example the number of simultaneous connections was set at a default of 5. Modify the REDIR command to include the following option.

```
REDIR /S:5/L:10  ;connection available for up to 5
                 servers and 10 simultaneous con-
                 nections.
```

Reboot the system and PC link will allow the user to connect up to 5 servers with up to 10 simultaneous connections.

APPENDIX C
INSITE LIBRARY PROGRAM

MS-KERMIT and ISIS KERMIT are available from:

Intel Corporation
2402 West Beardsley Road
Phoenix, Arizona 85027

ATTN: Insite User's Program Library

Telephone: (602) 869-3805

This is a public domain software and is available for a nominal charge of $25 each.

APPENDIX D
NDS-II/SERIES IV
TOOLBOX V2.0

NDS-II/SERIES-IV TOOLBOX V2.0

The NDS-II/Series-IV Toolbox V2.0 is a set of 7 diskettes with useful programs for NDS-II/Series-IV and OpenNET users. The programs used from this toolbox were MSCOPY.86, ReExporter, NET CONNECT. It contains many other useful utilities. An index listing the various programs in the Toolbox are listed below.

NDS-II/Series IV Toolbox 2.0